数理の窓

究極のプログラミング

既存のプログラム言語に物足りず、独自のプログラム言語を開発することは1つの“究極のプログラミング”だろう。プログラム言語は、処理内容を表す“ソースコード”の表記ルールと、そのソースコードをマシン語に翻訳するコンパイラのペアで構成される。

具体的には、四則演算、ループ処理、代入計算、再帰処理をどう“言語のように”表記するかのルールを定める。この時「理論上の完全なコンピューター[1)]」のすべての動作をカバーできることが必須だ。この性質は「チューリング完全」と呼ばれ、ほとんどの既存言語は満たしている。そして、別のプログラミング言語で書かれたソースでも、独自言語に翻訳できる必要もある。

一方、コンパイラの機能を処理順に並べると、

①ソースコードの字句の並びに分割＝字句解析
②字句が構文規則に合致するかチェック＝構文解析
③未定義変数利用違反などのチェック＝意味解析
④実行プログラムの効率向上のための最適化
⑤マシン語コードの生成

コンパイラ自体も1つのプログラムなので、実行のためマシン語に翻訳することが必要だ。これはブートストラップ問題と呼ばれるが、普通は、自力でマシン語に翻訳するか、コンパイラ自体を既存言語で書くことになる。

このように苦労して開発した独自言語を世に出すからには、表現力が豊か、人に理解しやすい、実行スピードが早い、特定のマシン構造に依存しない、といった特徴を持ちたい。ところが現実には難しい。マシン語に近くするとスピードは速いが、人が理解しづらくなる。文法ルールを厳格にすると安全性は上がるが、自由度は下がる。命令文を多く用意するとソースは短くなるが、メモリ管理などが難しくなる。つまり、各々の特徴はトレードオフ関係にあるのだ。優劣は設計思想に依存するため「究極のプログラム言語」はあり得ないとも言える。

求められる設計は、コンピューター初期のハードウェア、オープンソースの開発環境、3Dゲーム製作など環境により変わる。よって、プログラム言語たちの生存・栄枯盛衰の様は、生物の進化のようになるのも理解できる。

ところで、当然のことだがシステム構築のプロセスはコンパイラと似ている。優秀なコンパイラはユーザー要件を解析し、ニーズに合わせ優先事項を判断し、最適な“実行プログラム”を提供する。これを極めることも1つの究極のプログラミングといえるだろう。。

（外園 康智）

1）万能チューリングマシンと呼ばれる。世界中のコンピューターは“同等の計算機能”であり、万能チューリングマシン以上に複雑な計算のできる計算機は存在しないだろうというのが、一般的な見方である。